django-allauth-2fa Documentation

Release 0.4.3

Víðir Valberg Guðmundsson, Percipient Networks

Sep 15, 2023

Contents:

django-allauth-2fa adds two-factor authentication to django-allauth. django-allauth is a set of Django applications which help with authentication, registration, and other account management tasks.

Source code http://github.com/percipient/django-allauth-2fa

Documentation https://django-allauth-2fa.readthedocs.io/

CHAPTER 1

Features

- Adds two-factor authentication views and workflow to django-allauth.
- Supports Authenticator apps via a QR code when enabling 2FA.
- Supports single-use back-up codes.

CHAPTER 2

Compatibility

django-allauth-2fa attempts to maintain compatibility with supported versions of Django, django-allauth, and django-otp.

Current versions supported together is:

Django	django-allauth	django-otp	Python
4.1	0.53.0	1.2	3.8, 3.9, 3.10, 3.11
4.2	0.53.0	1.2	3.8, 3.9, 3.10, 3.11

CHAPTER 3

Contributing

django-allauth-2fa was initially created by Víðir Valberg Guðmundsson (@valberg), was maintained by Percipient Networks for many years, and is now maintained by Valohai. Please feel free to contribute if you find django-allauth-2fa useful!

1. Check for open issues or open a fresh issue to start a discussion around a feature idea or a bug.
2. If you feel uncomfortable or uncertain about an issue or your changes, feel free to email support@percipientnetworks.com and we will happily help you.
3. Fork the repository on GitHub to start making your changes to the **main** branch (or branch off of it).
4. Write a test which shows that the bug was fixed or that the feature works as expected.
5. Send a pull request and bug the maintainer until it gets merged and published.

3.1 Start contributing

Start by cloning the project with:

```
git clone https://github.com/valohai/django-allauth-2fa.git
```

The project uses hatch for building and package management. If you don't have hatch installed, you can do so by running:

```
pip install hatch
```

Setup you virtual environment with hatch:

```
hatch env create
```

3.2 Running tests

Tests can be run using pytest

```
hatch run pytest
```

3.3 Running the test project

The test project can also be used as a minimal example using the following:

```
hatch run python manage.py migrate
hatch run python manage.py runserver
```

3.3.1 Installation

Install *django-allauth-2fa* with pip (note that this will install Django, django-allauth, django-otp, qrcode and all of their requirements):

```
pip install django-allauth-2fa
```

After all the pre-requisities are installed, django-allauth and django-otp must be configured in your Django settings file. (Please check the django-allauth documentation and django-otp documentation for more in-depth steps on their configuration.)

```
INSTALLED_APPS = (
    # Required by allauth.
    'django.contrib.sites',

    # Configure Django auth package.
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',

    # Enable allauth.
    'allauth',
    'allauth.account',

    # Configure the django-otp package.
    'django_otp',
    'django_otp.plugins.otp_totp',
    'django_otp.plugins.otp_static',

    # Enable two-factor auth.
    'allauth_2fa',
)

MIDDLEWARE_CLASSES = (
    # Configure Django auth package.
    'django.contrib.auth.middleware.AuthenticationMiddleware',

    # Configure the django-otp package. Note this must be after the
    # AuthenticationMiddleware.
```

(continues on next page)

(continued from previous page)

```
    'django_otp.middleware.OTPMiddleware',

    # Reset login flow middleware. If this middleware is included, the login
    # flow is reset if another page is loaded between login and successfully
    # entering two-factor credentials.
    'allauth_2fa.middleware.AllauthTwoFactorMiddleware',
)

# Set the allauth adapter to be the 2FA adapter.
ACCOUNT_ADAPTER = 'allauth_2fa.adapter.OTPAdapter'

# Configure your default site. See
# https://docs.djangoproject.com/en/dev/ref/settings/#sites.
SITE_ID = 1
```

After the above is configure, you must run migrations.

```
python manage.py migrate
```

Finally, you must include the django-allauth-2fa URLs:

```
from django.urls import include, path

urlpatterns = [
    # Include the allauth and 2FA urls from their respective packages.
    path('accounts/two-factor/', include('allauth_2fa.urls')),
    path('accounts/', include('allauth.urls')),
]
```

Warning: Any login view that is *not* provided by django-allauth will bypass the allauth workflow (including two-factor authentication). The Django admin site includes an additional login view (usually available at `/admin/login`).

The easiest way to fix this is to wrap it in `staff_member_required` decorator and disallow access to the admin site to all, except logged in staff members through allauth workflow. (the code only works if you use the standard admin site, if you have a custom admin site you'll need to customize this more):

```
from django.contrib import admin
from django.contrib.admin.views.decorators import staff_member_required

# Ensure users go through the allauth workflow when logging into admin.
admin.site.login = staff_member_required(admin.site.login, login_url='/accounts/
↪login')
# Run the standard admin set-up.
admin.autodiscover()
```

3.3.2 Configuration

`ALLAUTH_2FA_TEMPLATE_EXTENSION`

Allows you to override the extension for the templates used by the internal views of django-allauth-2fa.

Defaults to `ACCOUNT_TEMPLATE_EXTENSION` from allauth, which is `html` by default.

This can be used to allow a different template engine for 2FA views.

ALLAUTH_2FA_ALWAYS_REVEAL_BACKUP_TOKENS

Whether to always show the remaining backup tokens on the Backup Tokens view, or only when they're freshly generated.

Defaults to `True`.

ALLAUTH_2FA_REMOVE_SUCCESS_URL

The URL name to redirect to after removing a 2FA device.

ALLAUTH_2FA_SETUP_SUCCESS_URL

The URL name to redirect to after setting up a 2FA device.

ALLAUTH_2FA_FORMS

Used to override forms, for example: `{'authenticate':   'myapp.forms.TOTPAuthenticateForm'}`

Possible keys (and default values):

- `authenticate`: `allauth_2fa.forms.TOTPAuthenticateForm`
- `setup`: `allauth_2fa.forms.TOTPDeviceForm`
- `remove`: `allauth_2fa.forms.TOTPDeviceRemoveForm`

3.3.3 Advanced Configuration

Forcing a User to Use 2FA

A `User` can be forced to use 2FA based on any requirements (e.g. superusers or being in a particular group). This is implemented by subclassing the `allauth_2fa.middleware.BaseRequire2FAMiddleware` and implementing the `require_2fa` method on it. This middleware needs to be added to your `MIDDLEWARE_CLASSES` setting.

For example, to require a user to be a superuser:

```
from allauth_2fa.middleware import BaseRequire2FAMiddleware

class RequireSuperuser2FAMiddleware(BaseRequire2FAMiddleware):
    def require_2fa(self, request):
        # Superusers are require to have 2FA.
        return request.user.is_superuser
```

If the user doesn't have 2FA enabled they will be redirected to the 2FA configuration page and will not be allowed to access (most) other pages.

3.3.4 Changelog

0.12.0 - Unreleased

Possibly breaking changes

- You can't write to *allauth_2fa.app_settings* variables anymore; instead modify the underlying *django.conf.settings* settings.

New features

- Add flag to make the required entry of an otp code for device removal optional (#169)

0.11.1 - July 13, 2023

Patch release to address a packaging issue where templates weren't included (#176, #177).

0.11.0 - July 4, 2023

We didn't wait one year from the last release for this on purpose, I swear!

Minimum dependency versions

- The minimum version of Python for this release is 3.7.
- The minimum version of Django for this release is 3.2.
- The minimum version of django-otp for this release is 1.1.x.
- The minimum version of django-allauth for this release is 0.53.0.

Possibly breaking changes

- The *token* field in forms is now *otp_token*; if you have subclassed forms, or are using custom templates, this may require adjustment.

New features

- Allow customizing view success URLs via app_settings
- Show secret on setup page to allow for non-QR-code devices
- You can customize the QR code generation arguments in TwoFactorSetup (#156)
- 2FA can be disabled using backup tokens (#155)
- You can now override the forms used by the views using settings, like allauth does (#161)

Infrastructure

- The package now has (partial) type annotations.

0.10.0 - July 4, 2022

If you're using a custom template for the 2FA token removal view, note that you will need to also display the `token` field beginning with this version (PR #135 in particular).

The minimum version of django-otp was bumped to 0.6.x.

- Update CI bits and bobs by @akx in #137
- Drop support for django-otp 0.5.x by @akx in #138
- Require user token to disable 2FA by @SchrodingersGat in #135

0.9 - April 11, 2022

This release dropped support for Python 3.5 and added support for Django 4.0.

- Improves documentation for protection Django admin with 2FA. Contributed by @hailkomputer in #91.
- Autocomplete on the token entry form is disabled. Contributed by @qvicksilver in #95.
- Stop restricting a class of an adapter in *TwoFactorAuthenticate* by @illia-v in #96
- Use same base template as upstream allauth by @ErwinJunge in #98
- Redirect to next, when given via GET or POST by @ErwinJunge in #99
- Allow TOTP removal when no backup device is present by @akx in #126
- Fix for subclassed OTP adapter by @squio in #129
- Replace Travis with GitHub Actions by @akx in #110
- Drop EOL Python 3.5, modernize for 3.6+ by @akx in #106
- Remove Django 1.11 from tox, and add Django 4.0b1. by @valberg in #118
- Typo in docs by @beckedorf in #122
- Add pre-commit, and run it. by @valberg in #121
- Rename master -> main by @akx in #123
- Declarative setup.cfg by @akx in #124
- Use Py.test for tests + fix coverage reporting by @akx in #127
- Require2FAMiddleware improvements by @akx in #107
- Miscellaneous housekeeping by @akx in #130

0.8 February 3, 2020

- Drop support for Python 2.7 and Python 3.4.
- Officially support Python 3.7 and 3.8.
- Drop support for Django 2.0 and Django 2.1.
- Officially support Django 3.0.

0.7 September 10, 2019

- Remove more code that was for Django < 1.11.
- Officially support Django 2.0 and Django 2.1.
- Officially support django-otp 0.7.
- Do not include test code in distribution, fix from @akx, PR #67.
- Support for more complex user IDs (e.g. UUIDs), fix from @chromakey, see issue #64 / PR #66.
- The extension used by the 2FA templates is customizable. Originally in PR #69 by @akx, split into PR #71.
- The QR code is now included inline as an SVG instead of being a separate view. PR #74 by @akx.
- A new mixin is included to enforce a user having 2FA enabled for particular views. Added in PR #73 by @akx.
- Passing additional context to the `TwoFactorBackupTokens` was broken. This was fixed in PR #73 by @akx.
- A configuration option (`ALLAUTH_2FA_ALWAYS_REVEAL_BACKUP_TOKENS`) was added to only show the static tokens once (during creation)> PR #75 by @akx.

0.6 February 13, 2018

- Drop support for Django < 1.11, these are no longer supported by django-allauth (as of 0.35.0).

0.5 December 21, 2017

- Avoid an exception if a user without any configured devices tries to view a QR code. This view now properly 404s.
- Redirect users to configure 2FA is they attempt to configure backup tokens without enabling 2FA first.
- Add base middleware to ensure particular users (e.g. superusers) have 2FA enabled.
- Drop official support for Django 1.9 and 1.10, they're no longer supported by the Django project.
- Added Sphinx-generated documentation. A rendered version is available at.

0.4.4 March 24, 2017

- Adds trailing slashes to the URL patterns. This is backwards compatible with the old URLs.
- Properly support installing in Python 3 via PyPI.

0.4.3 January 18, 2017

- Adds support for forwarding `GET` parameters through the 2FA workflow. This fixes `next` not working when logging in using 2FA.

0.4.2 December 15, 2016

- Reverts the fix in 0.4.1 as this breaks custom adapters that inherit from `OTPAdapter` and *don't* override the `login` method.

0.4.1 December 14, 2016

- Fixed a bug when using a custom adapter that doesn't inherit from `OTPAdapter` and that overrides the `login` method.

0.4 November 7, 2016

- Properly continue the allauth login workflow after successful 2FA login, e.g. send allauth signals
- Support using `MIDDLEWARE` setting with Django 1.10.
- Support customer `USERNAME_FIELD` on the auth model.

0.3.2 October 26, 2016

- Fix an error when hitting the TwoFactorBackupTokens view as a non-anonymous user.

0.3.1 October 5, 2016

- Properly handle an `AnonymousUser` hitting the views.

0.3 October 5, 2016

- Support custom `User` models.
- Fixed a bug where a user could end up half logged in if they didn't complete the two-factor login flow. A user's login flow will now be reset. Requires enabled the included middle: `allauth_2fa.middleware.AllauthTwoFactorMiddleware`.
- Disable autocomplete on the two-factor code input form.
- Properly redirect anonymous users.
- Minor simplifications of code (and inherit more code from django-otp).
- Minor updates to documentation.

0.2 September 9, 2016

- Add tests / tox / Travis support.
- Don't pin dependencies.
- Officially support Django 1.10, drop support for Django 1.7.

0.1.4 May 2, 2016

- Autofocus the token input field on forms.

0.1.3 January 20, 2016

- Fix deprecation notices for Django 1.10.

0.1.2 November 23, 2015

- Fixed an error when a user enters invalid input into the token form.

0.1.1 October 21, 2015

- Project reorganization and clean-up.
- Added support for Microsoft Authenticator.
- Support being installed via pip.
- Pull more configuration from Django settings (success URL).
- Support disabling two-factor for an account.

0.1 April 4, 2015

- Initial version by Víðir Valberg Guðmundsson

CHAPTER 4

Indices and tables

- genindex
- modindex
- search